实验 3　加法器与 ALU

2020 年秋季学期

杨飞洋 191220138

目录

1. 实验目的：

实现一个带有逻辑运算的 ALU，增强对 verilog 中模块实例化的理解，加深并熟练 verilog 中的各种语法，为之后的实验做好准备工作。

2. 实验原理（知识背景）

加法是数字系统中最常执行的运算，加法器是 ALU 的核心部件。减法可以看作是被减数与取负后的减数进行加法。即用加法器同时实现加法和减法两种运算。乘法也可以利用移位相加的算法来实现。因此，加法器可以说是计算机中最“繁忙”的部件了。
4 位补码表示十进制数范围是-8 ~ 7
加法运算遵循补码加法规则。
减法通过取减数的补码再进行加法运算。相应的对进位位和溢出位做一些处理。
下图是 ALU 的功能表

功能选择	功能	操作
000	加法	A+B
001	减法	A-B
010	取反	Not A
011	与	A and B
100	或	A or B
101	异或	A xor B
110	比较大小	If A>B then out=1; else out=0;
111	判断相等	If A==B then out=1; else out=0;

逻辑运算就是每一位对应的进行运算，算术运算遵循补码的运算法则。并且加减运算时需要考虑进位位 C 和溢出位 overflow.在逻辑运算时，C 和 overflow 默认为 0。

3. 实验环境/器材

系统：windows10
开发软件：Quartus 17.1 Lite
开发板：DE10 Standard
仿真环境：ModelSim
芯片：Cyclone V ， 5CSXFC6D6

4. 设计思路

先是加法运算，将 16 个数分成两组，[-8,-1]即[1000,1111]为一组，[0,7]即[0000,0111]为二组。
如果两个数在不同组，必不会溢出，需要考虑会不会产生进位
如果两个数都在一组，必然会出现溢出和进位，result = A + B – 16,例如
-1 + -2 = 1111 + 1110 – 16 = 1101 = -3
如果两个数都是在二组，若结果 <= 7，直接输出，如果 > 7 则溢出，并输出一个负的结果

Verilog 中 module 模块代码：

```
if(A <= 7 && B <= 7)begin
   if(A + B > 7)begin
      over = 1;
   end
   result = A + B;
end
else if(A >= 8 && B >= 8)begin
   over = 1;
   C = 1;
   result = A + B - 16;
end
else begin
   if(A + B > 15)begin
      result = A + B - 16;
      //over = 1;
      C = 1;
   end
   else result = A + B;
end
```

再是减法：

减法器不仅仅只是将减数取反加一在进行加法运算就好了。

进位位 C 比较容易判断，只要 A < B，就会有进位

如果 A 和 B 在同组，必定不会有溢出。

如果在不同组，则可能溢出，根据测试，做出相应的判断即可。

```
if(A < B)
   C = 1;
else C = 0;
if(A >= 8 && B >= 8)begin
   if(A + (~B+1) > 15)begin
      result = A + (~B+1) - 16;

   end
   else
      result = A + (~B+1);
end
else if(A <= 7 && B <= 7)begin
   if(A >= B)
      result = A - B;
   else begin

      result = A + (~B+1);
      end
end
else if(A >= 8 && B <= 7)begin
   result = A - B;
   if(A < 8)
      over = 1;
end
else begin
   result = A + ~B + 1;
   if(result > 7)
         over = 1;
end
```

接下来的取反、与、或、异或、判断相等相对比较容易，

010 取反：result = \~{}A

011 与：result = A \& B

100 或：result = A | B

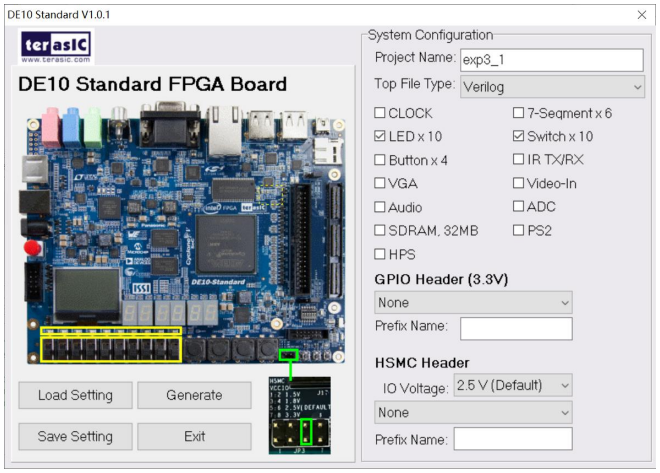
101 异或：result = (~A&B) | (A&~B)

还有一个比较功能，假如 A、B 在不同组，则在二组的显然是大的，如果都在二组或都在一

组，直接比较大小得出结果即可。

5.实验步骤

一．利用 systembuilder 建立工程



二．编写 Verilog 代码

在工程目录打开 qpf 文件
新建一个.v 文件，写一下 ALU 的模块

```
module ALU(A,B,switch,result,out,over,C);
```

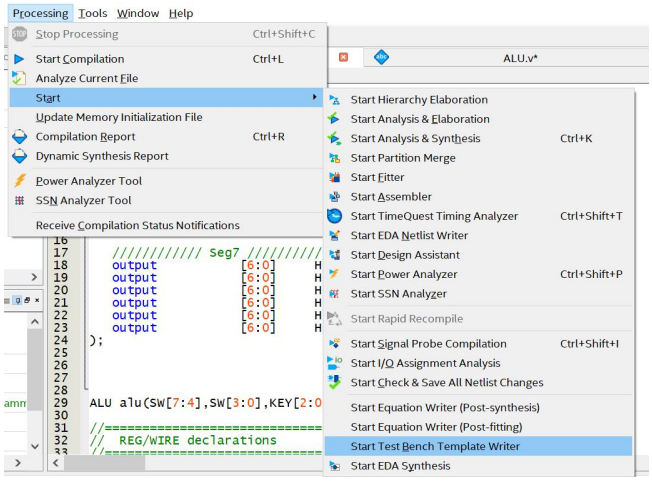
再将该文件名更改为 ALU.v,（文件名与模块名一致）
在顶层文件中根据需要调用 ALU 模块

```
ALU alu(SW[7:4],SW[3:0],KEY[2:0],LEDR[3:0],LEDR[9],LEDR[4],LEDR[5]);
```

进行编译调试，直至成功。

三．编写 test bench

利用 processing 中的 start test bench template writer 让系统自动生成 test bench 测试文件

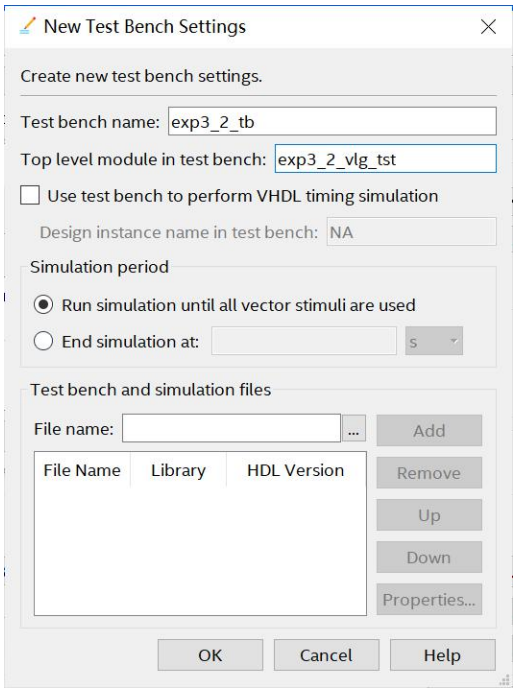


在 simulation 中的 modelsim 文件中打开.vt 文件

修改仿真时间单位，`timescale 10 ns/ 1 ps
在 initial 下方 begin 和 end 之间写入测试数据
为避免风险，将 always 语句注释。

四．设置 test bench

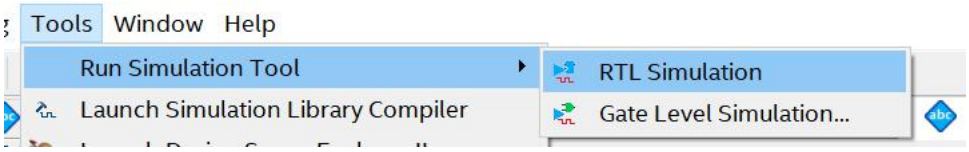
在 assignments->settings->simulation->compile test bench 中设置 test bench，new 一个 test bench，注意将顶层实体修改为 exp3_2_vlg_tst



再将 simulation->modelsim 中的.vt 文件 add 进去

五．仿真测试

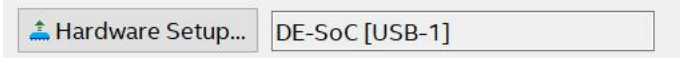
利用 RTL Simulation 调用 modelsim



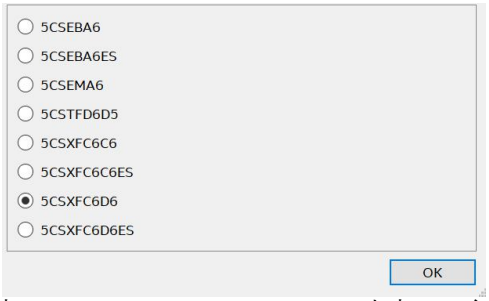
在波形图中 zoom full 就可以看到测试结果

六．利用 fpga 测试

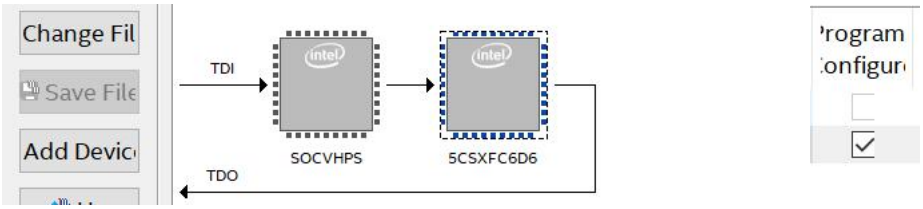
检查工程目录下有 sof 文件
将 fpga 充电，usb 接电脑，点击 programmer 按钮 
hardware 选择 usb



select device 选择倒数第二个



选中 5CSXFC6D6，change file，选中.sof 文件，接着在 program 上打勾，在点击 start 开始测试



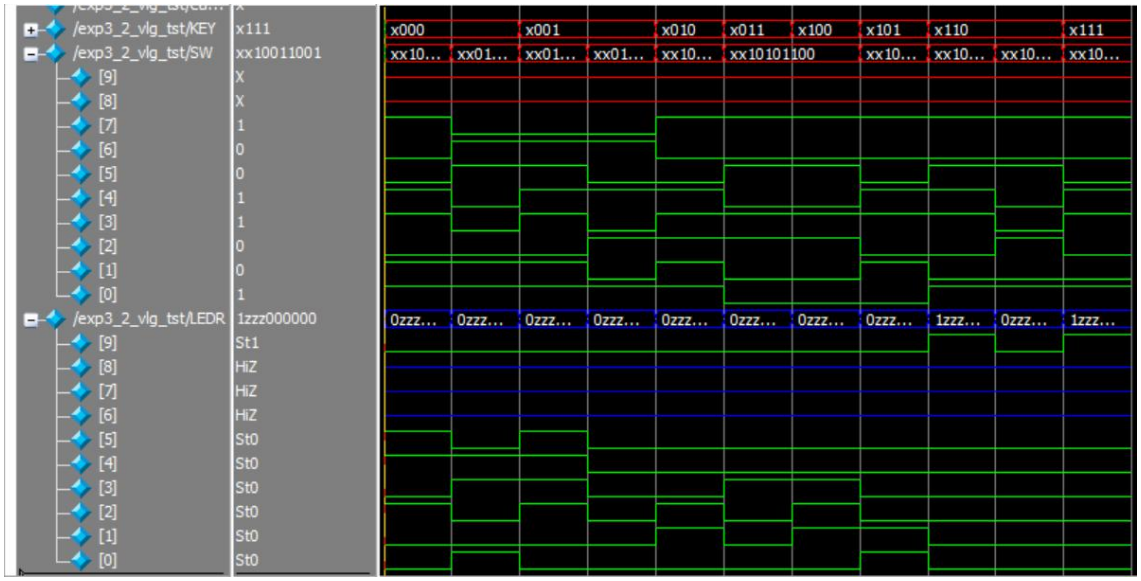
6.测试方法

加减法测试：取几组具有代表性的数据，比如 5 + (-5) , 4 + 3 , (-8) + (-6) , -8 – 7 , 7 – (-4) , 2 - 4 等，涵盖正常运算、溢出状态、进位状态，尽量全面测试
下面是测试文件代码：

```
initial
begin
// code that executes only once
// insert code here --> begin
KEY[2:0] = 3'b000;SW[7:0] = 8'b10011011;#20;
KEY[2:0] = 3'b000;SW[7:0] = 8'b01100011;#20;
KEY[2:0] = 3'b001;SW[7:0] = 8'b01111011;#20;
KEY[2:0] = 3'b001;SW[7:0] = 8'b01010101;#20;
KEY[2:0] = 3'b010;SW[7:0] = 8'b10011111;#20;
KEY[2:0] = 3'b011;SW[7:0] = 8'b10101100;#20;
KEY[2:0] = 3'b100;SW[7:0] = 8'b10101100;#20;
KEY[2:0] = 3'b101;SW[7:0] = 8'b10011010;#20;
KEY[2:0] = 3'b110;SW[7:0] = 8'b10111001;#20;
KEY[2:0] = 3'b110;SW[7:0] = 8'b10100101;#20;
KEY[2:0] = 3'b111;SW[7:0] = 8'b10011001;#20;
// --> end
$display("Running testbench");
end
```

7.实验结果

仿真结果：



下载运行结果：

由于拍出来也看不大清 button 的按键情况，所以我认为拍照的证明效果意义不大，还是应该由现场查验来做。

8.实验中遇到的问题

一．需要点击两次的 start test bench template writer，才能生成 test bench 测试文件。

二 .一开始一位减法只要把减数取反加一在进行加法就好了，后来发现事情并没有这么简单。还需要对 C 和 overflow 进行操作。

三．在 systembuilder 中还选择了七段数码管，后来发现好像不是很合适。就让他放在那没管。

四．修改完代码之后，开发板上没有相应的改变，是因为需要重新编译才能对.sof 文件进行纠正。

9.实验得到的启示

一．不能有侥幸心理，如果仿真都没有通过，开发板是必不会成功的。

二．实验前应该先想好思路，把想法搞清楚，不能稀里糊涂的直接写代码，会导致很多的 bug。

三．通常来说，波形图正确了，开发板上的结果就都是正常的。

四．本来其实对补码运算的概念不是很清楚，这个实验强迫我学会了这个，也对补码表示整数有了更深的理解。

10.意见和建议

一．希望多一些测试样例，给一些数据。
二．希望多一些关于 verilog 语言模块化编程的指导。

思考题

1. 应该比较 A、B 和结果的符号位。
2. 不一样，如果是减法，应该先取反加一在加法，不应该是先取反，再做(A + 1) + ~B，举个例子，比如此时 cin = 1 减法，B = 0，如果只取反不加一，则变成 A + 1111 + 1，可能发生不应该的溢出，所以需要先取反加一。
   方法二是正确的。

3，一元约简运算较好，位运算比比较器高效。